

L Number	Hits	Search Text	DB	Time stamp
5	4	((("5971524") or ("6145961") or ("6155668") or ("6196736"))).PN.	USPAT	2003/03/20 11:57
6	2	((("5563886") or ("6157650"))).PN.	USPAT	2003/03/20 12:25
7	1	"6529522"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 12:26
8	1	("6529522").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 12:26
9	2205	(ito near masamichi).in. (koji near takahashi).in. (hiroki near yamashita).in.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 12:31
10	0	((ito near masamichi).in. (koji near takahashi).in. (hiroki near yamashita).in.) and (cannon).as.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 12:31
11	0	((ito near masamichi).in. (koji near takahashi).in. (hiroki near yamashita).in.) and (cannon)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 12:31
12	760	((ito near masamichi).in. (koji near takahashi).in. (hiroki near yamashita).in.) and canon	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 13:36
13	2	((("6011784") or ("5953350"))).PN.	USPAT	2003/03/20 13:21
14	58	(imag\$6 with process\$6) and (scanner\$5 camera) and (printer\$5) and (network adj interface\$6) and (function near9 (description describ\$5))	USPAT	2003/03/20 13:59
15	4	(imag\$6 with process\$6) and (scanner\$5 camera) and (printer\$5) and (network adj interface\$6) and (function near9 (description describ\$5)) and (device\$5 near9 class\$5)	USPAT	2003/03/20 13:40
16	158	(imag\$6 with process\$6) and (network adj interface\$6) and (function near9 (description describ\$5))	USPAT	2003/03/20 14:01
17	58	(exchang\$6 negotiat\$6) with (imag\$6 near9 process\$6) with function\$6	USPAT	2003/03/20 14:12
18	0	6298164.uref.	USPAT	2003/03/20 14:13
19	6	((("5970221") or ("5826017") or ("5732198") or ("5710908") or ("5706410") or ("5689255"))).PN.	USPAT	2003/03/20 14:14
20	1	6298164.pn. and (program\$5 cod\$6 softwar\$6)	USPAT	2003/03/20 14:15
21	1	6298164.pn. and (program\$5 function\$5 descript\$6 class\$5 cod\$6 softwar\$6)	USPAT	2003/03/20 14:16
24	21	jetsend	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/03/20 15:21
26	2	("6202096" "6298164").pn. and (program\$3 cod\$6)	USPAT	2003/03/20 16:59
27	2	("6202096" "6298164").pn. and (nic network card\$3)	USPAT	2003/03/20 15:34
28	63	(ieee adj "1394") with (network adj interface\$5)	USPAT	2003/03/20 15:35
29	7	(ieee adj "1394") with (network adj interface\$5 adj card\$5)	USPAT	2003/03/20 15:36
30	7	(ieee adj "1394") with (nic (network adj interface\$5 adj card\$5))	USPAT	2003/03/20 15:48

31	12	(program\$3 cod\$6) same (interaction adj protocol\$5)	USPAT	2003/03/20 16:04
32	673	(program\$3 adj cod\$6) same encod\$6	USPAT	2003/03/20 16:06
33	49	(physical\$5) with (udp rmtpt)	USPAT	2003/03/20 17:00
34	6	(physical\$5) with (session near protocol\$5)	USPAT	2003/03/20 17:59
35	0	6298164.pn. and (prefer\$6)	USPAT	2003/03/20 18:50
36	116	(nic (network adj interface adj card\$3)) with (tcp/ip ieee)	USPAT	2003/03/20 19:11
37	1	(nic (network adj interface adj card\$3)) with (provid\$6 fast\$5) with communicat\$6 with (tcp/ip ieee)	USPAT	2003/03/20 19:13
38	33	(nic (network adj interface adj card\$3)) same (provid\$6 fast\$5) same (tcp/ip ieee)	USPAT	2003/03/20 19:33
39	4286	(nic (network adj interface adj card\$3))	USPAT	2003/03/20 19:46
40	510	gateway with storage\$5	USPAT	2003/03/20 19:46
41	299	gateway near9 storage\$5	USPAT	2003/03/20 19:54
42	1	("5970221").PN.	USPAT	2003/03/20 19:54
43	1	6298164.pn. and stack\$5	USPAT	2003/03/20 20:01
-	1	("6108727").PN.	USPAT	2003/03/19 15:26
-	8	jetsend	USPAT	2003/03/20 20:01
-	7	jetsend and (program\$5 cod\$5 function\$5 negotiat\$6)	USPAT	2003/03/19 15:37
-	0	jetsend and (convert\$6 same function\$5)	USPAT	2003/03/19 15:38
-	3	jetsend and (convert\$6 same (image\$5 data))	USPAT	2003/03/19 15:38
-	5	jetsend and (convert\$6 compress\$6 decompress\$6)	USPAT	2003/03/19 15:40
-	2	jetsend and (convert\$6 and (compress\$6 decompress\$6))	USPAT	2003/03/19 15:40
-	2	jetsend and ((convert\$6 convert\$6) and (compress\$6 decompress\$6))	USPAT	2003/03/19 15:42
-	2	jetsend and (chang\$6 same (color\$5 image\$5))	USPAT	2003/03/19 15:46
-	1	("5214779").PN.	USPAT	2003/03/19 16:15
-	2216	(program\$5 code\$3) same (image with process\$5 with function\$5)	USPAT	2003/03/19 16:18
-	47	(program\$5 near2 code\$3) same (image with process\$5 with function\$5)	USPAT	2003/03/19 16:25
-	1	(program\$5 code\$3) same (image with process\$5 with function\$5) same (export\$6) same (scanner\$3 camera) same (printer\$5)	USPAT	2003/03/19 16:26
-	1	(program\$5 code\$3) same (imag\$6 with process\$5 with function\$5) same (export\$6) same (scanner\$3 camera) same (printer\$5)	USPAT	2003/03/19 16:27
-	62	(program\$5 code\$3) same (imag\$6 with process\$5 with function\$5) same (scanner\$3 camera) same (printer\$5)	USPAT	2003/03/19 16:29
-	0	((program\$5 code\$3) same (imag\$6 with process\$5 with function\$5) same (scanner\$3 camera) same (printer\$5)) and cannon.as.	USPAT	2003/03/19 16:28
-	39	((program\$5 code\$3) same (imag\$6 same process\$5 samefunction\$5) same (scanner\$3 camera) same (printer\$5)) and (cannon hp hewkett)	USPAT	2003/03/19 16:32
-	5	((program\$5 code\$3) same (imag\$6 same process\$5 same function\$5) same (scanner\$3 camera) same (printer\$5)) and (negotiat\$6)	USPAT	2003/03/19 16:33
-	250	((program\$5 code\$3) same (imag\$6 same process\$5 same function\$5) same (scanner\$3 camera) same (printer\$5))	USPAT	2003/03/19 16:39
-	1	("6157650").PN.	USPAT	2003/03/19 16:40

[Advanced Groups Search](#) [Preferences](#) [Groups Help](#)

jetsend exchange negotiate

Google Search

Groups search result 1 for jetsend exchange negotiate**Sun Microsystems Reseller** • Low Prices. Fast Delivery. New/Used Spare Part & Storage
OnLine • www.mcac.com

Sponsored Links:

Sun Reseller - TC Digital • Find new, used and refurbished equipment here. Large
inventory. • www.tcdigital.com**Cisco Catalyst Switches** • Cisco Premier Partner Great Prices. Friendly Help. • www.tribecaexpress.com

From: ~KJ~ (No Spamkjcambra@sprynet.com)

Search Result 1

Subject: Jini Jousts

Newsgroups: [comp.os.inferno](#)

This is the only article in this thread

Date: 1999/01/28

View: [Original Format](#)

[Note - repost due to system crash
Table info. at bottom of article]

#####

Sun's Jini jousts with rivals

By Stephen Shankland
Staff Writer, CNET News.com
January 25, 1999, 7:00 a.m. PT

The race is on to create the universal fabric that could supplant today's ungainly networking technology, as Sun is set to debut its technology called Jini.

Several companies, believing today's networking too cumbersome or limited, are working on technologies that connect everything from light switches to supercomputers in one ubiquitous network. The idea is to reap the benefits of ever-broader networks without having to deal with obtuse, unwieldy technology.

But as often happens with high-technology efforts in their infancy, big-name companies will compete to establish their own vision of a universal network, thereby creating some confusion along the way.

Many of the biggest names in computing are scrambling to set the standards and line up supporters including Sun Microsystems, Microsoft, Hewlett-Packard, IBM, and Lucent.

After several previews, Sun will formally unveil its Jini technology in San Francisco today. Several companies, including networking software maker Novell, have licensed the Java-based technology, which allows the "spontaneous networking" of devices. And network hardware maker Cisco demonstrated a Jini-powered cable modem earlier this month, and Sun executives showed a free-standing Jini hard disk from Quantum in December.

Sun says Jini will let traveling businessmen easily plug into hotel printers, let parents at work peek through child-monitoring cameras at home, and let people turn up the air conditioning before they get home. When a Jini-enabled device plugs into a network, it automatically announces itself and its capabilities.

Meanwhile, Sun arch rival Microsoft, eyeing the approach of Jini, hastened to take the wraps off its Universal Plug and Play initiative at the Consumer Electronics Show earlier this month. Microsoft's technology is an extension of the Plug and Play hardware recognition system introduced with Windows 95,

but it will let people tie together devices without needing a computer. With it, devices announce themselves and their capabilities when plugged into a network.

"We're looking at this with the view that all kinds of devices not currently networked today are going to want to be networked," said Alec Saunders of Microsoft's intelligent appliances division. Universal Plug and Play will work with "smart objects" such as light switches or volume controls, intelligent appliances such as Web-enabled telephones, or computers.

Microsoft has found support from Compaq, Intel, ATI, 3Com, AMD, Kodak, and others.

But Sun designed Jini from scratch, and therefore unlike Microsoft isn't hampered by having to support the "ancient" architecture of the PC, said Gartner Group analyst David Smith.

"I see Universal Plug and Play as an evolution of technology, an incremental improvement to an OK solution. I see Jini as something that is much more elegant, part of the overall movement to network computing and ubiquitous devices," Smith said.

Sun and Microsoft have similar intentions for their technologies: Both companies believe their network technologies will drive sales of more traditional products such as operating systems and servers.

Microsoft criticized Java as an out-of-the blue technology where Universal Plug and Play uses existing Internet communications standards and registration services. "We're leveraging a big heritage of existing technologies, bringing Internet technologies into a new class of devices," Saunders said.

Jim Waldo, Jini's chief architect, though, spurned Universal Plug and Play as a mere initiative that checks all the boxes on the "buzzword bingo card." It's far behind Jini, which has had beta software out for months and will be launched as a product Monday, Waldo said.

Microsoft says Universal Plug and Play devices won't need to have a computer on the network, but will be able to take advantage of one if it's there.

Jini is designed to bypass computers altogether, Waldo said: "All we require is Java someplace on the network."

Farther out into the future, Microsoft's research arm is working on another networking project, an operating system called Millennium that lets computers share tasks across a network, automatically adjusting to new components being added or removed.

One Millennium prototype called "Borg" is a Java virtual machine that can make a cluster of computers look like a single one when running Java programs. Another prototype, called Continuum is similar, but adds support for other programming languages such as Visual Basic, C, and C++.

Sun and Microsoft have the most prominent efforts, but they aren't the only big names in the business of redefining how networks happen. HP, Lucent, and IBM all have plans of their own.

HP's JetSend

Hewlett-Packard introduced its JetSend technology in 1997 as a way to shield users from the complexities of different document formats, said HP's Kipp Martel. The technology complements both Universal Plug and Play as well as Jini, Martel said.

The technology lets two devices negotiate the best way to share documents so, for example, a JetSend-enabled printer could accept an image from a JetSend-enabled digital camera. The two devices communicate to figure out what common format will preserve the image quality best. JetSend also could let a cable TV operator send video on demand over a network without having to worry about what devices the viewer will use.

While the company has deployed JetSend in its own printers and, more recently, scanners as well, HP will next move JetSend into computers this spring, Martel said. This will allow the computer to take over JetSend communications for non JetSend-enabled devices, he said.

JetSend is the Esperanto of the computing world, Martel said, promising "universal viewability," Martel said. HP's JetSend-enabled CapShare 910 hand-held scanner transfers scanned images to computers using Adobe's cross-platform Portable Document Format (PDF).

JetSend has been licensed by several companies, including Panasonic, Minolta, Siemens, Xerox, and Canon.

HP will support CapShare both for Jini and Universal Plug and Play, even though it costs more to support two platforms, Martel said. "We are committed to this vision of a world of seamless connectivity," he said.

Lucent's Inferno

Lucent's Inferno effort gives equipment such as smart phones, Internet appliances, or set-top boxes a small-footprint operating system that can connect to networks or run programs within a virtual machine. The system currently supports programs written in Lucent's Limbo language or Sun's stripped-down version of Java called PersonalJava.

Inferno was publicly announced in 1997, but Smith has seen little progress since. "It's a fantastically well-designed piece of work. It's very applicable in today's world of the Internet, it's just that Lucent has not been able to market it," Smith said.

Lucent declined to comment for this story, but said Inferno can be used in a lot of the same ways as Jini. "Right now we're in the process of working on applications and technology related to specific markets, but it would be premature to comment on that now," said spokeswoman Barbara McClurken. Lucent expects to make a new Inferno announcement in "a couple of months," she said.

IBM's T Spaces

IBM's research wing, meanwhile, is working on a Java-based technology called T Spaces that lets computers, digital assistants, and other devices share data such as email or database queries.

The technology complements Jini and helps to achieve the common goal of "pervasive computing." However, T Spaces is only one of IBM's projects from its research labs that applies to that future world.

An IBM paper notes that T Spaces "basically connects all things to all things," and can run on very small devices. The technology would make it easy for resources such as printers, scanners, fax machines, or software services to be shared across networks with lots of different kinds of computers.

All these different technologies have appeal, Gartner Group's Smith said, but are likely to show up first in homes and small offices. It's hard to manage systems of thousands of devices attached to networks, he said.

But as the systems catch on, they'll likely "trickle up" to higher levels as the networking technologies become more robust and pervasive, he said.

"There is a tremendous desire to move us toward easier-to-use networking," Smith said.

Intel has a technology too which it is promoting as the "Home Network API" in an effort to develop a common method for computerized control of home devices and Sun and Philips are providing a similar technology called HAVi.

(from Table)

Sun - Jini

Lets any Jini-enabled device or software module share services for "spontaneous networking" with other Jini devices.

Microsoft - Plug & Play

Extends hardware recognition beyond PCs to let electronic devices easily connect without needing a computer.

Microsoft - Millennium

Lets collections of computers automatically divvy up computing tasks across network.

IBM - T-Spaces

Java-based system that allows computers, palm computers, or other devices to share messages, database queries, print jobs, or other network services.

Lucent - Inferno

Small-footprint network operating system to let smart phones, set-top boxes, and other equipment plug into the network and run Java or other programs.

HP - JetSend

Communication technology that lets networked devices such as printers, scanners, or digital cameras negotiate common file formats for data exchange.

#####

remove "No_Spam" from addr. to reply directly

[Google Home](#) - [Advertise with Us](#) - [Search Solutions](#) - [Services & Tools](#) - [Jobs, Press, & Help](#)

US-PAT-NO: 6298164

DOCUMENT-IDENTIFIER: US 6298164 B1

TITLE: PCL conversion of JETSEND images

----- KWIC -----

JETSEND is a device-to-device communications protocol for local and wide area networks, that allows network devices to intelligently negotiate information exchange between them, without the need of involving a separate computer to negotiate the conversation. The JETSEND protocol allows the two devices to connect over a network, negotiate the best possible data type, provide device status, and exchange information, all without intervention from a user or from a third device such as a computer to negotiate the communication between the two devices. JETSEND protocol is described at length in "HP JETSEND.TM. Communications Technology, Section I: Architectural Overview; Section II: Protocol References; and Section III: E-Material Specification", Version 1.0, Hewlett-Packard Company, 1997. These documents are incorporated by reference as if set forth in full herein.

As an example, a JETSEND-enabled scanner (or other image input device such as a digital camera) can capture image information and then send the image information directly to a JETSEND-enabled printer (or other output device such as a facsimile or a projector) at a remote network location. According to the JETSEND protocol, the scanner would send information about its capabilities to the printer. Such information would include, for example, the available formats, color capabilities, bit length, and image resolution, of images that the scanner could acquire. In response, the printer would return information concerning the format in which the image information should be sent. The image information is captured by the scanner and thereafter sent directly to the printer for output. All such communications occur without the intermediary of a user or a computer such as a server to negotiate the communication, thereby enabling the scanner to communicate directly with the printer.

One problem currently being encountered with JETSEND concerns modification of legacy systems, which do not incorporate JETSEND protocol, into JETSEND-enabled systems. For example, current networked printers represent an enormous installed base of network printers that are not currently JETSEND-enabled.

US-PAT-NO: 6298164

DOCUMENT-IDENTIFIER: US 6298164 B1

TITLE: PCL conversion of JETSEND images

----- KWIC -----

a **program** memory for storing process steps executable to perform (a) a monitoring step to monitor control byte information as the compressed image data is being stored to a buffer, so as to determine when the end of a scan line has been reached; (b) a modifying step, responsive to the end of a scan line being reached, to modify the control byte of the final portion of compressed image data in the buffer so as to correspond to exactly one scan line; (c) a transmitting step to transmit image data in the buffer prefixed by a printer description language command that includes a byte count of the number of bytes in the transmission; and (d) a starting step to start a new scan line in the buffer beginning with any un-transmitted data together with a control byte that specifies either a repeat count or a byte count in correspondence to whether the un-transmitted image data was run length encoded or a literal representation, and followed by newly-received compressed image data; and

a processor for executing the process steps stored in said **program** memory.

10. A computer-readable medium for storing computer-executable **program code**, said computer-executable **program code** for converting compressed image data into a compressed raster image in units of a scan line, the compressed image data including mixed portions of run-length encoded representations in which a first byte is a control byte that specifies a repeat count and a second byte is image data to be repeated, mixed together with literal representation portions in which a first byte is a control byte that specifies a byte count of uncompressed image data bytes that follow, said computer-executable **program code** comprising:

11. Computer-readable **program code** according to claim 10, further comprising:

12. Computer-readable **program code** according to claim 11, wherein modification comprises flipping black and white bits of the compressed image data.

[Advanced Groups Search](#) [Preferences](#) [Groups Help](#)

jetsend exchange negotiate

Google Search

[Web](#) [Images](#) [Groups](#) [Directory](#) [News](#)

Related groups

[comp.os.inferno](#)Searched Groups for **jetsend exchange negotiate**.

Results 1 - 1 of 1. Search took 0.14 seconds.

Sorted by relevance [Sort by date](#)

Jini Jousts

... **JetSend** Communication technology that lets networked devices such as printers, scanners,

or digital cameras **negotiate** common file formats for data **exchange**. ...

[comp.os.inferno](#) - Jan 28, 1999 by ~KJ~ - [View Thread \(1 article\)](#)

Sponsored Links

Negotiation Tips

Get everything you want in your next negotiation - only \$19.95

[negotiate.i-tips.net](#)

Interest: _____

[See your message here...](#)

jetsend exchange negotiate

Google Search

[Google Home](#) - [Advertise with Us](#) - [Search Solutions](#) - [Services & Tools](#) - [Jobs, Press, & Help](#)

This is Google's cache of http://www.unet.univie.ac.at/aix/aixprggd/progcomc/ch8_rpc.htm.

Google's cache is the snapshot that we took of the page as we crawled the web.

The page may have changed since that time. Click here for the [current page](#) without highlighting.

To link to or bookmark this page, use the following url: http://www.google.com/search?q=cache:4Vz0beaY8EAJ:www.unet.univie.ac.at/aix/aixprggd/progcomc/ch8_rpc.htm+remote+procedure+call&hl=en&ie=UTF-8

Google is not affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **remote procedure call**

[[Next Article](#) | [Previous Article](#) | [Book Contents](#) | [Library Home](#) | [Legal](#) | [Search](#)]

Communications Programming Concepts

Chapter 8. Remote Procedure Call

Remote Procedure Call (RPC) is a protocol that provides the high-level communications paradigm used in the operating system. RPC presumes the existence of a low-level transport protocol, such as Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP), for carrying the message data between communicating programs. RPC implements a logical client-to-server communications system designed specifically for the support of network applications.

This chapter provides the following information about programming RPC:

- [RPC Model](#)
- [RPC Message Protocol](#)
- [RPC Authentication](#)
- [RPC Port Mapper Program](#)
- [Programming in RPC](#)
- [RPC Features](#)
- [RPC Language](#)
- [rpcgen Protocol Compiler](#)
- [List of RPC Programming References](#)

The RPC protocol is built on top of the eXternal Data Representation (XDR) protocol, which standardizes the representation of data in **remote** communications. XDR converts the parameters and results of each RPC service provided.

The RPC protocol enables users to work with **remote** procedures as if the procedures were local. The **remote procedure** calls are defined through routines contained in the RPC protocol. Each **call** message is matched with a **reply** message. The RPC protocol is a message-passing protocol that implements other non-RPC protocols such as batchin and broadcasting **remote** calls. The RPC protocol also supports callback procedures and the **select** subroutine on the server side.

A *client* is a computer or process that accesses the services or resources of another process or computer on the network. A *server* is a computer that provides services and resources, and that implements network services. Each network *service* is a collection of **remote** programs. A **remote** program implements **remote** procedures. The procedures, their parameters, and the results are all documented in the specific program's protocol.

RPC provides an authentication process that identifies the server and client to each other. RPC includes a slot for the authentication parameters on every **remote procedure call** so that the caller can identify itself to the server. The client

package generates and returns authentication parameters. RPC supports various types of authentication such as the UNIX and Data Encryption Standard (DES) systems.

In RPC, each server supplies a program that is a set of **remote** service procedures. The combination of a host address, program number, and **procedure** number specifies one **remote** service **procedure**. In the RPC model, the client makes a **procedure call** to send a data packet to the server. When the packet arrives, the server calls a dispatch routine, performs whatever service is requested, and sends a reply back to the client. The **procedure call** then returns to the client.

The RPC interface is generally used to communicate between processes on different workstations in a network. However, RPC works just as well for communication between different processes on the same workstation.

The Port Mapper program maps RPC program and version numbers to a transport-specific port number. The Port Mapper program makes dynamic binding of **remote** programs possible.

To write network applications using RPC, programmers need a working knowledge of network theory. For most applications, an understanding of the RPC mechanisms usually hidden by the **rpcgen** command's protocol compiler is also helpful. However, use of the **rpcgen** command circumvents the need for understanding the details of RPC.

[[Next Article](#) | [Previous Article](#) | [Book Contents](#) | [Library Home](#) | [Legal](#) | [Search](#)]

[Advanced Search](#) [Preferences](#) [Language Tools](#) [Search Tips](#)

remote procedure call

Google Search

[Web](#) - [Images](#) - [Groups](#) - [Directory](#) - [News](#)Searched the web for **remote procedure call**.Results **11 - 20** of about **1,120,000**. Search took **0.09** second**Free Virus Scan - Sobig.f**

Sponsored Links

www.Stop-Sign.com If your computer is slowing down or crashing, free scan. Kills worms.**Citations: Lightweight remote procedure call - Bershad, Anderson ...**

B. Bershad, T. Anderson, L. Lazowska, and H. Levy. Lightweight remote procedure call. ... Lightweight remote procedure call. In Proc. ...

citeseer.nj.nec.com/context/854/0 - 34k - [Cached](#) - [Similar pages](#)

Sponsored Links

Remote Procedure ShutdownIs Caused by a Virus. We Offer Live Help by Phone. 1-888-GEEK-HELP
www.888GeekHelp.com

Interest:

[See your message here...](#)**Citations: Remote Procedure Call - Nelson (ResearchIndex)**

BJ Nelson. Remote Procedure Call. Technical Report CMU--81--119, Carnegie-Mellon University, 1982. 62 citations found. ... Nelson, BJ Remote procedure call. ...

citeseer.nj.nec.com/context/11878/0 - 61k - [Cached](#) - [Similar pages](#)[\[More results from citeseer.nj.nec.com \]](#)**RFC 1050 (rfc1050) - RPC: Remote Procedure Call Protocol ...**

... RFC 1050 - RPC: Remote Procedure Call Protocol specification. ... 3. THE RPC MODEL The remote procedure call model is similar to the local procedure call model. ...

Description: RPC: Remote Procedure Call Protocol Specification. Sun Microsystems. April 1988.

Category: [Computers](#) > [Internet](#) > [RFCs](#) > [1001 - 1100](#)www.faqs.org/rfcs/rfc1050.html - 49k - Sep 5, 2003 - [Cached](#) - [Similar pages](#)[\[More results from www.faqs.org \]](#)**Remote Procedure Call (RPC) - [Translate this page]**

Remote Procedure Call (RPC). L'appel de fonction est ... L'idée de Remote Procedure Call n'est pas nouvelle. De fait, de nombreux systèmes ...

www.crevola.com/francois/rpc.html - 18k - [Cached](#) - [Similar pages](#)**Chapter 8. Remote Procedure Call**

Chapter 8. Remote Procedure Call. Remote Procedure Call (RPC) is a protocol that provides the high-level communications paradigm used in the operating system. ...

www.unet.univie.ac.at/aix/aixprgdd/progcom/ch8_rpc.htm - 7k - [Cached](#) - [Similar pages](#)**Tweakzone Forum - remote procedure call**[Index / Besturingssystemen / remote procedure call](#). ... [Index / Besturingssystemen](#)/ [remote procedure call](#). FastTopics v3.0 beta is hosted by Grafix.nl.forum.tweakzone.nl/topic/6777 - 40k - [Cached](#) - [Similar pages](#)**Remote Procedure Call**

Windows 2000 Network Architecture. Remote Procedure Call (RPC) is an interprocess communication technique to allow client and server software to communicate. ...

www.microsoft.com/windows2000/techinfo/reskit/en-us/cnet/cnad_arc_plgn.asp - 13k - Sep 5, 2003 - [Cached](#) - [Similar pages](#)[\[More results from www.microsoft.com \]](#)**RPC - a whatis definition - see also: Remote Procedure Call**

Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer in a network without having ...

www.whatis.com/definition/0,289893,sid9_gci214272,00.html - 38k - Sep 5, 2003 - [Cached](#) - [Similar pages](#)**Microsoft Vulnerability May Affect Off-Campus Users**

... Alternatively, you can download the Remote Procedure Call security patch from the official Microsoft download site or the BevoWare download site. ...

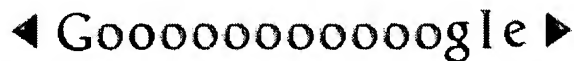
www.utexas.edu/its/alerts/dhsalert07252003.html - 27k - Sep 5, 2003 - [Cached](#) - [Similar pages](#)

Microsoft Remote Procedure Call (RPC) Server Vulnerability

INFORMATION BULLETIN. L-126: Microsoft **Remote Procedure Call** (RPC) Server Vulnerability.

[Microsoft Security Bulletin MS01-041]. July 30, 2001 17:00 GMT. ...

www.ciac.org/ciac/bulletins/l-126.shtml - 8k - Sep 5, 2003 - [Cached](#) - [Similar pages](#)



Result Page: [Previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [Next](#)

[Search within results](#)

[Google Home](#) - [Advertise with Us](#) - [Business Solutions](#) - [Services & Tools](#) - [Jobs, Press, & Help](#)

©2003 Google


Embedded.com
Thinking inside the box

 Where do you go to build
your Silicon Strategy?

**EE TIMES
NETWORK**

E-CYCLOPEDIA

HOME

REGISTER

ABOUT US

ADVERTISING

FEEDBACK

 Embedded Systems
Conference
San Francisco

Search

 Search the EE Times
Network

PRODUCT NEWS
**Toshiba doubles single-die
NAND capacity**

Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

**TimeSys JVM makes
predictable Java**

The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News >

DATELINE: EUROPE

**IMEC and National
Semiconductor collaborate**

IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe >
EMBEDDED.COM LINKS

- ▶ Embedded Systems Conferences
- ▶ Embedded Systems Programming Magazine
- ▶ Embedded Systems Europe
- ▶ Downloadable Code
- ▶ Product Demos
- ▶ Internet Resources
- ▶ Industry Events
- ▶ Buyer's Guide
- ▶ Site Map

COMPANY STORE

- ▶ CD-ROM

Embedded Systems

PROGRAMMING

Internet Appliance Design

An Introduction to the JetSend Protocol

John Meadows

JetSend is a media-independent communications protocol that can exchange information in its proper context without any product-specific knowledge or device drivers. This article provides an overview of the JetSend protocol.

JetSend is a media-independent communications protocol, developed by Hewlett-Packard, that provides interoperability among a wide variety of devices in differing vertical markets. You may be thinking, "Do we really need another communications protocol?" The answer is yes, and here is the reason: a need exists for a protocol that supports device-to-device content exchange. True, protocols already exist for point-to-point data exchange (FTP, IrTranP, and so on), but the problem with these protocols is that they don't fully describe the content being exchanged, and as a result the data transferred is both machine dependent and OS dependent.

The fact that JetSend is able to exchange information in its proper context, without any product-specific knowledge or device drivers, gives it tremendous value. Increasingly, embedded devices are required to exchange data with other devices directly (that is, no PC will be required). With JetSend, the user of a device doesn't need to know what protocol is being used to transfer data. These devices might be digital cameras, cell phones, PDAs, printers, scanners, or set-top boxes, to name

 Free Cypress
USB Net
Seminar.

click here

 Cypress
EZ-USB FX2

 Integrated
USB 2.0
MCUs
for high
bandwidth
applications.

 CYPRESS
Connecting From Last Mile
to First Mile

- Embedded Books
- The Work Circuit

NetSeminar
Services

Learn more about
designing PCBs with
bare-die components
and the benefits of
OLE automation

March 26

EE TIMES NETWORK

Online Editions

[EE TIMES](#)
[EE TIMES ASIA](#)
[EE TIMES CHINA](#)
[EE TIMES FRANCE](#)
[EE TIMES GERMANY](#)
[EE TIMES KOREA](#)
[EE TIMES TAIWAN](#)
[EE TIMES UK](#)

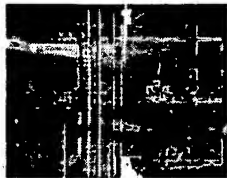
Web Sites

- [CommsDesign](#)
- [GaAsNET.com](#)
- [iApplianceWeb.com](#)
- [Microwave Engineering](#)
- [EEdesign](#)
- [Deepchip.com](#)
- [Design & Reuse](#)
- [Embedded.com](#)
- [Elektronik i Norden](#)
- [Planet Analog](#)
- [Semiconductor Business News](#)
- [The Work Circuit](#)
- [TWC on Campus](#)

ELECTRONICS GROUP SITES

- [ChipCenter](#)
- [EBN](#)
- [EBN China](#)
- [Electronics Express](#)
- [NetSeminar Services](#)
- [QuestLink](#)
- [Custom Magazines](#)

PRODUCT CATALOGS



Motorola Smart Networks
Resource Guide 2003

NETWORK RESOURCES

Ⓜ JOB SEARCH

JD

printers, scanners, or set-top boxes, to name just a few. Another compelling reason for using JetSend is that it already has support in many of the devices just listed. All of Hewlett-Packard's standalone printers and scanners support JetSend. Also, JetSend printer support for Windows CE PDAs-although not native-can be obtained via download from HP. This article will attempt to provide an overview of the JetSend protocol.

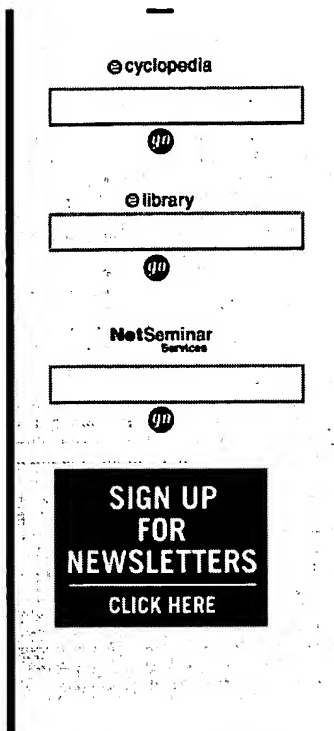
Overview of JetSend

The HP JetSend protocol is a peer-to-peer communication protocol. The protocol allows two devices to connect, negotiate data types, and exchange information. With this new protocol comes a new paradigm for job control in embedded devices. For example, say a person wants to print a picture from a digital camera. The current job control model requires the user to transfer the picture to a printer through a two-step process. First the user must launch a compatible digital camera application on a PC to receive the desired pictures from the camera. Then a host application-possibly the same one-prints the desired pictures using a compatible printer driver. In this job model, the host computer acts as a translator between two devices that speak different languages.

The JetSend job control model differs in that it is a one-step, peer-to-peer communication process. A host PC no longer controls the exchange of data between two devices, which removes the need to load a special driver on the sending device because both devices speak a common language. One side effect, however, is that the host PC can no longer be relied upon for any data translation.

Within JetSend, all data exchanged between devices is accomplished using surfaces. Each surface object has a name, a description, and content. The description is analogous to a file header, and contains information about the type and content of the surface. The content portion may be null, data, or a reference to another surface (a child surface). The data contained in a surface is called electronic material (e-material).

As an example, a two-page document, one with image and text and the second with just an image, can be represented by six surfaces, as shown in [Figure 1](#). The encodings of the surfaces are also shown. In a printing example, the **vAssociation** encoding describes the entire



Copyright © 2003 CMP Media

LLC

[Privacy Statement](#)

document or print job. The **vPlane** encoding controls the layout of pages to be printed. The **vImage** and **vText** encoding describe the content of the image and text.

In this example the "Text1" child surface on the first page of the document is a text description of the "Image1" picture. This description might be something like "Billy's first birthday-January 01, 1999." A raster image of this text string is also offered in case the receiving device does not support the **vText** encoding. In other nonprinting applications, surface encodings may take on different meanings.

Surface interaction model: the rules of the language

JetSend is a layered protocol designed to be machine independent as well as transport independent. The components of the protocol shown in [Figure 2](#) are as follows:

- The device-specific code is the application itself
- The Interaction Policies control the method of interaction between devices by enforcing certain types of policies. One of the policies-the Job Policy-will be covered in detail in the section on using the JetSend API
- The e-material routines allow the device-specific code to format data to be communicated in a standard JetSend format that another JetSend enabled device will be able to understand. Surfaces are e-material
- The JetSend Interaction Protocol (JIP) manages the sending and receiving of surfaces
- The JetSend Session Protocol (JSP) is responsible for managing a reliable data connection with a remote device
- Transport Independent (TI) layer decouples the JetSend stack from the OS-dependent network layer

Data sent by the application layer (device-specific code) must progress down through each layer before being sent out over a transport medium and then back up through the JetSend stack of the receiving device.

Machine independence is achieved through the use of e-material, which specifies data types and formats that do not rely on software language data types. An example of this would

be the e-material definition of a simple integer type. The first byte of the type definition is a value type identifier, which tells what type of integer it is, as shown in [Figure 3](#). The byte(s) that follows is the actual value of the integer in network byte order (big endian-most significant byte first).

The TI layer is responsible for providing a generic API to the non-JetSend transport protocols. Operating system-specific code is needed to translate generic TI API messages into OS network driver calls and vice versa.

Using the JetSend API

To communicate using JetSend, device-specific code must interact with an API composed of three main sections: the Activity Manager, Interaction Policies, and e-material routines.

Activity manager API . The Activity Manager is the heart of the JetSend API and fulfills two main functions. First, the Activity Manager is responsible for managing JetSend sessions. A session is a reliable full duplex communication channel established between two JetSend-enabled devices. All user data transferred from one device to another is conveyed over one or more sessions. The second main function is event handling. The Activity Manager manages events coming from the lower layers of the JetSend stack by providing an event queue and polling mechanism to process and free events. [Listing 1](#) shows the routines that make up Activity Manager API. More about how the Activity Manager works will be discussed in the section on sending application data.

JetSend interaction policies . The second main section of the JetSend API is concerned with supporting one or more of the JetSend Interaction Policies. This component of the JetSend Protocol defines various typical sequences of interactions that devices agree to follow. Most of the toolkits available implement the session policy and a portion of the job policy. Altogether five policies have been defined as follows:

- Session policy-how to establish and disconnect sessions between devices
- Job policy-how to send documents between senders and receivers
- Self policy-how to exchange global information about a device, such as label,

icon, and passwords

- Status policy-how to give status about a device and about jobs
- Address policy-how to program devices with new destination addresses

Of the five policies, the session policy is supported directly by the Activity Manager through the use of session management routines. The session policy defines how sessions are created and destroyed. In the session policy, roles for both passive and active devices are defined. The passive device acts as a consumer and the active device acts as a producer. In order for two devices to communicate, one device must begin listening for another device to connect to it on a given transport channel. The listening device opens a passive session and waits for notification of a connecting device.

The remaining policies-job, self, status, and address-define how data is formatted and exchanged between devices.

The job policy is used to send and receive application data over JetSend sessions. Two methods for exchanging data, the PUSH method and the PULL method, are defined by the job policy. As the name implies, the PUSH method is used to push data from a sending device to a receiving device, as would be done for printing, faxing, and so on. I will give an example of a PUSH sender/receiver in the section on sending application data. The PULL method is used when the receiving device needs to be able to select certain pieces of data from a sending device, as would be the case for some type of monitoring device. An example of this might be a JetSend-enabled digital Walkman that is able to download selected songs from a jukebox-like storage device using the PULL method.

The self policy, status policy, and address policy together provide the mechanisms to convey device attribute and status information. The self policy provides the ability for a remote device to obtain device attribute information. This information includes the device's address, name (in text and in a Windows icon bitmap), and PIN number to allow support for user authentication. The self policy is often used in conjunction with the address policy.

The status policy, as the name implies, allows a device to obtain the status of another device. Using this policy it is possible to report conditions like paper jam, low battery, and so

on.

The last policy in this grouping, the address policy, defines a mechanism to convey addressing information about additional devices available on a network. The JetSend protocol doesn't have a mechanism to automatically discover JetSend devices on a given network, so the address of at least one JetSend device must be known in order to use JetSend to accomplish a given task. Users program a device with the addresses of the devices they wish to use. For point-to-point transports like IrDA, this is as simple as programming a single default address into the device. On network transports like IP, the address policy can be used to program devices with additional addresses. The address policy allows a network administrator to set up a well-known device address on a network that holds the addresses of other JetSend devices. Once addresses are obtained, the type and capabilities of a device can be discovered by using the self policy.

E-material routines . The e-material routines are used to create and parse surfaces in order to intelligently exchange data. E-material data structures have predefined data elements and data types that include values, strings, and lists. Using e-material routines, simple data types like integers and strings are converted to e-material types to build surface descriptions. E-material surface descriptions are constructed to fully describe user data and list possible encodings in which the sending device can transfer the data. This gives the receiving device the ability to choose an encoding that best suits its devices' capabilities. The process of selecting a data transfer format is called **device negotiation** . To ensure that devices of the same class will be able to exchange data, a default data format is defined. For example, the default encoding for image devices is 300 x 300 dpi, monochrome, RLE compressed raster data. All JetSend-certified image devices (printers, scanners, copiers, cameras, whiteboards, and so on) must be able to send and/or receive this encoding (Section 2.2.3 of the JetSend Protocol Specification, v. 1.5). The ability to add new encodings while enforcing support for a default encoding will allow JetSend to maintain backwards compatibility. You have a device like a digital Walkman that supports audio clips in MP3 format. The sending device might support new audio formats while still supplying the old digital Walkman with MP3 audio clips. The new digital Walkman can thus take advantage of a new format, while the old devices-having only

the old format-are still supported.

Sending data

The sample PUSH sender given in [Listing 2](#) will be used to describe the steps required to transfer user data from one device to another (see also [Figure 5](#)). At the beginning of the routine **RunSimpleSender()** the Activity Manager, which is responsible for managing sessions, is initialized by calling **jamInitialize()**. This sets up an event queue inside the Activity Manager that can be polled by the device-specific code using the function **jamGetNextEvent()**. This polling mechanism allows the device-specific code to process events that have propagated up through the JetSend stack. Next, the PUSH sender is initialized by calling **psInitialize()**. Upon initialization the PUSH sender installs callbacks for events it will handle as part of the job policy (shown graphically in [Figure 4](#)). To communicate from one JetSend-enabled device to another, an active JetSend session must be created, which is accomplished by calling **jamStartSession()**. The transport address is specified in this call. In this example IrDA is selected as the transport. For a session to succeed in opening, a JetSend receiver must be listening for a session on the same network channel selected in the **jamStartSession()** call.

When a session is first established (that is, the session opens successfully) the PUSH receiver constructs an "IN" surface and sends it over the newly opened session. The purpose of the IN surface is to inform the connecting device that a receiver is available which supports the job policy. Upon receiving the remote device's IN surface, the PUSH sender posts a **psInArrivedEvent** event to the Activity Manager. The sending device-specific code then reads this event via **jamGetNextEvent()** and begins the process of sending a document.

To illustrate how a document is sent, let's once again use the document in [Figure 2](#). The **vAssociation** surface is constructed using routines from the e-material library and then sent out over the open session to the remote device. The PUSH receiver, called by the Activity Manager via a callback routine, posts a **prJobStartEvent** when the **vAssociation** surface is received. The remote device code, in processing this event, parses the **vAssociation** surface description and determines what portion of the data it wishes to receive.

For this example we will assume that the receiving device will ask that the entire document be sent through a series of requests to the sending device. First the receiver requests the first **vPlane** surface (or page) in our document. The sender receives this request as a **psSurfaceRequestEvent** from the **jamGetNextEvent()** routine. A description of the data contained in the requested **vPlane** must then be constructed via e-material routines and sent to the remote device. The **vPlane** description defines how a page is laid out. In our sample document this description would contain (x,y) coordinates of both the image and text objects. The receiving device then knows where to place the objects for display or printing. At this point the receiver has the knowledge that the current job consists of a two-page document, and that the first page has both an image object and a text object on it. So far, no user data (content) has been transferred.

In the next step, the sending device requests the **vImage** surface which contains a description of the image data. This description is in the form of a list of possible encodings supported by the sender. The receiving device reads the list of data encodings and requests that the content be sent in a format that best suits its device capabilities. The sender-upon receiving a request-sends content messages until the entire body of the **vImage** is sent. The remote device then requests the **vText** surface be sent. Our example document has the text content as part of the **vText** object on page one, so once the **vText** surface is sent, the first page is fully transferred and can be displayed, printed, or stored. Since there are two **vPlanes** in the **vAssociation** surface of our example document, the remaining **vPlane** will be requested using the same techniques.

When all the data is sent, the session can be closed or left open. A session would be left open if additional data were to be transferred. For example, the sending side could have additional documents (or jobs) that it desires to send. The additional documents would be sent in the same manner as I previously described. The possibility also exists that the receiver would want to get updates of any changes made to a received document by the sending device. An example of this might be a JetSend-enabled display connected to a portable computing device.

A few good reasons

The JetSend protocol is well suited for embedded devices for a couple of reasons. First, the protocol has been kept simple by defining everything in terms of surfaces. Therefore, a typical implementation of the JetSend stack on an embedded device will be somewhere around 60K of code. The second reason is that JetSend is peer to peer, and will allow devices to communicate with a range of other devices, a feature which will likely become a requirement as the number and usage of new devices increase. The third reason why JetSend is a good fit for embedded devices is the negotiation aspect of the protocol. Negotiation allows embedded devices to support a small minimum set of data formats that will be directly used by the device. For example an image class device does not have to support TIFF, JPEG, GIF, Windows bitmap, and so on, but only needs to be able to support raster image data in at least the minimum 300dpi RLE encoding. As long as memory is not free, the negotiation feature of JetSend will allow embedded devices to communicate with a wide variety of other devices at a reasonable cost.

So far I have covered only a few of the possible uses of JetSend. Hewlett-Packard continues to develop the protocol and has added JetSend support to many of their printers and scanners. New e-material encodings such as vCard and vCalendar are currently being added to the specification to enable the exchange of contact and calendar information. A new interaction policy, the control policy, has been developed to support remote control of JetSend-enabled devices. Also added to the specification is support for proxies. As an example, an e-mail proxy can be used to provide e-mail support for devices that do not support e-mail, but do support sending to proxies. To learn more about these new features, as well as for additional information on JetSend, visit www.jetsend.com.

John Meadows is a member of the technical staff in the E-Services Practice at Questa Corp. He has almost 14 years of experience in the industry, focused primarily on networking and communication protocols. Prior to joining Questa, he worked as a software specialist for Redcom Laboratories which designs and manufactures Telecom switching equipment for hostile environments. John has an ASEE from Alfred State College and a BSCS from the Rochester Institute of Technology.

FIGURES & LISTINGS

Figure 1: Two-page document with images and text

Figure 2: The JetSend protocol stack

Figure 3: JetSend Integer encodings

Figure 4: The PUSH sender installs callbacks for events it will handle as part of the job policy

Figure 5: Jet Send transaction

Listing 1: Activity Manager API routines

Listing 2: Sample PUSH sender

Electronics Marketplace**• SEGGER's emWin: a Royalty-Free Graphics Library**

SEGGER's emWin is the ONLY professional embedded GUI package available that can compile under C and C++, making emWin the most compact and portable graphics library today. For more details, call 603-929-6435.

• The Fastest Embedded Processor Ever - Xtensa V

Test drive Tensilica's Xtensa V processor, the fastest ever according to EEMBC Certification Labs.

• Quality PCB & Thick Film Hybrid Circuits @Low Cost

High-quality PCB and thick film circuits at low cost. From prototypes to high-volume quantities, multilayers. ISO9002 certified manufacturing. Assembly and engineering service. Online quote in milliseconds.

• PEG (Portable Embedded GUI) Graphics Library/Tools


PEG is a GUI Development Package comprised of a full set of tools for embedded systems. PEG comes with a complete class library, runs fast, is small in size and completely rommable. PEG is delivered with C++ source code and is ROYALTY FREE!

• Embedded HW&SW Design Firm

Choose Cornerstone Design Group, Inc. as a design partner for your next embedded hardware and firmware project.


Buy a link NOW:

**EE TIMES INTERNET USAGE STUDY**




Embedded.com

Thinking inside the box



Check out our buyers guide
on EEdesign.com.



[E-CYCLOPEDIA](#)
[HOME](#)
[REGISTER](#)
[ABOUT US](#)
[ADVERTISING](#)
[FEEDBACK](#)

Embedded Systems
Conference
San Francisco

Search

Search the EE Times
Network

PRODUCT NEWS

Toshiba doubles single-die NAND capacity

Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

TimeSys JVM makes predictable Java

The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News

DATELINE: EUROPE

IMEC and National Semiconductor collaborate

IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe

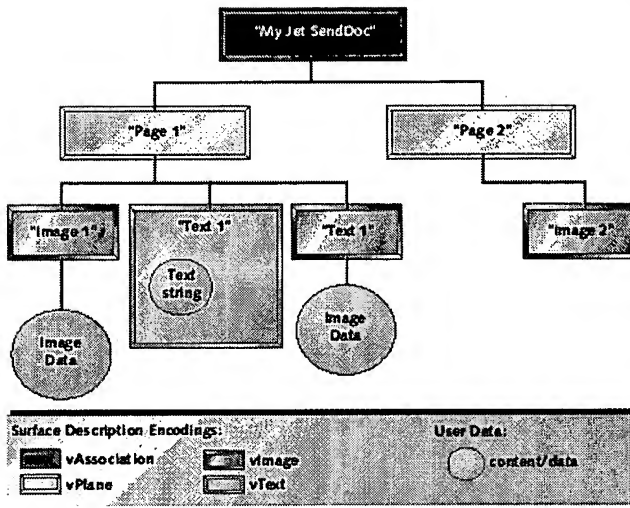
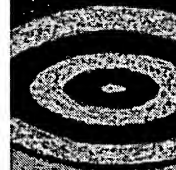



Figure 1: Two-page document with images and text

[Back](#)

Is Your
Career
on
Target?

Search
thousands
of jobs,
get career advice
join our
mentor boards

Visit
TheWorkCircuit.com
today!
[click here](#)



Embedded.com
Thinking inside the box

**semiconductor
BUSINESS NEWS**

Full News Coverage
of the Semi Industry

[E-CYCLOPEDIA](#)
[HOME](#)
[REGISTER](#)
[ABOUT US](#)
[ADVERTISING](#)
[FEEDBACK](#)

Embedded Systems Conference San Francisco

Search

Search the EE Times Network

PRODUCT NEWS

Toshiba doubles single-die NAND capacity
Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

TimeSys JVM makes predictable Java
The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News >

IMEC and National Semiconductor collaborate
IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe >



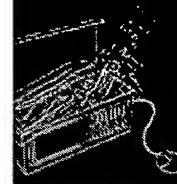
Figure 2: The JetSend protocol stack


[Back](#)

Embedded Systems Conference San Francisco

April 22-26, 2003
Moscone Center

146 classes and full-day tutorials, 90 industry experts





Embedded.com
Thinking inside the box

semiconductor
BUSINESS NEWS

Full News Coverage
of the Semi Industry

[E-CYCLOPEDIA](#)
[HOME](#)
[REGISTER](#)
[ABOUT US](#)
[ADVERTISING](#)
[FEEDBACK](#)

Embedded Systems Conference San Francisco

Search

Search the EE Times Network

PRODUCT NEWS

Toshiba doubles single-die NAND capacity
Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

TimeSys JVM makes predictable Java
The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News >

IMEC and National Semiconductor collaborate
IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe >

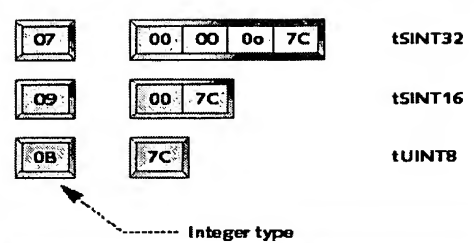



Figure 3: JetSend integer encodings


[Back](#)



**EE TIMES
INTERNET
USAGE
STUDY**

Share YOUR
THOUGHTS and
INSIGHTS
on how
you use the
INTERNET.

**EE TIMES
NETWORK**



Embedded.com
Thinking inside the box

**semiconductor
BUSINESS NEWS**

Full News Coverage
of the Semi Industry

E-CYCLOPEDIA
HOME
REGISTER
ABOUT US
ADVERTISING
FEEDBACK

Embedded Systems Conference San Francisco

Search

Search the EE Times Network

PRODUCT NEWS

Toshiba doubles single-die NAND capacity
Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

TimeSys JVM makes predictable Java
The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News ▶

DATELINE: EUROPE

IMEC and National Semiconductor collaborate
IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe▶

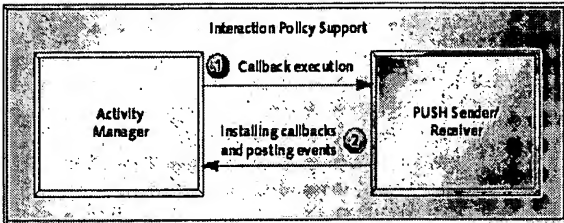
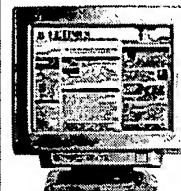



Figure 4: The PUSH sender installs callbacks for events it will handle as part of the job policy


[Back](#)



**EE TIMES
INTERNET
USAGE
STUDY**


Share YOUR THOUGHTS and INSIGHTS on how you use the INTERNET.






Embedded.com

Thinking inside the box



Check out our buyers guide
on EEdesign.com.



[E-CYCLOPEDIA](#)
[HOME](#)
[REGISTER](#)
[ABOUT US](#)
[ADVERTISING](#)
[FEEDBACK](#)

Embedded Systems
Conference
San Francisco

Search

Search the EE Times
Network

PRODUCT NEWS

Toshiba doubles single-die NAND capacity

Toshiba has introduced a 2Gb single-die NAND flash memory with double the capacity of the company's present largest single-die NAND flash memory.

TimeSys JVM makes predictable Java

The JTime series of software development kits are the first embedded Java development environments to comply with the real-time specification for Java.

More Product News >

DATELINE: EUROPE

IMEC and National Semiconductor collaborate

IMEC and National Semiconductor are jointly developing a 0.18-micron and follow-up generation of Silicon Germanium (SiGe)-based BiCMOS process technology optimized for low-power applications.

More News From Europe

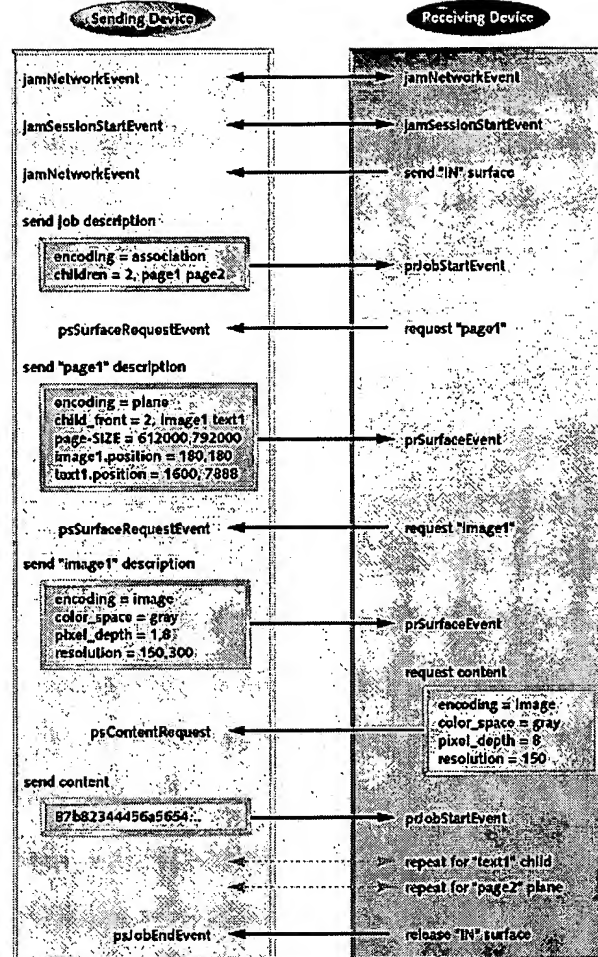
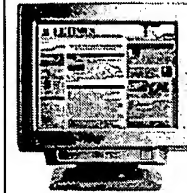


Figure 5: Jet Send transaction

[Back](#)



EE TIMES INTERNET USAGE STUDY

Share YOUR
THOUGHTS and
INSIGHTS
on how
you use the
INTERNET.

EETIMES
NETWORK

US-PAT-NO: 6298164

DOCUMENT-IDENTIFIER: US 6298164 B1

TITLE: PCL conversion of JETSEND images

----- KWIC -----

FIG. 3 illustrates the architecture of software for an image input device such as scanner 104, as stored in and executed by the NIC. Such software is stored as computer-executable process steps in a computer readable medium, such as the aforementioned EEPROM 16. As shown in FIG. 3, the architecture of the software extends through the NIC between the LAN and the networked input device, so as to provide interface for the networked device to the LAN. Thus, the architecture of the software includes physical layer 31, plurality of different protocol stacks 32, JETSEND interaction protocol 34, and device-specific applications such as scanner control application 35. Device-specific applications are applications concerning the functionality of the network device, such as the aforementioned scanner control application 35.

Plural protocol stacks 32 are preferred since they increase the flexibility of a networked device by allowing it to communicate via the network interface on the LAN using plural different network protocols. In the FIG. 3 example, two different network protocol stacks are shown, including IPX protocol stack 36 and IP protocol stack 37. Other protocol stacks may also be provided, such as a DDP protocol stack, a UDP protocol stack, a NetBIOS protocol stack, or an AppleTalk protocol stack. Of course, it is possible for a networked device to operate over even a single protocol stack.

FIG. 4 illustrates the architecture of software for an image output device such as printer 115, as stored in and executed by the NIC for the printer. In much the same way as the software for image input devices, such software is stored as computer-executable process steps in a computer readable medium, such as in EPROM 16. As shown in FIG. 4, the architecture of the software extends through the NIC between the LAN and the networked output device, so as to provide interface for the networked device to the LAN. Thus, the architecture of the software in FIG. 4 includes physical layer 41, a plurality of different protocol stacks 42, device-specific applications such as printing control applications 44, a JETSEND interface protocol 45, and a JETSEND agent 46. The

NIC further includes a printer interface module such as XP module 47. XP module 47 provides process steps for a standard interface between the NIC and the printer. In particular, and as illustrated in FIG. 4, the printer includes a print engine and a print controller, and XP module 47 interfaces directly to the print controller over whatever interface is provided therebetween, such as the aforementioned bi-directional interfaces or the aforementioned uni-directional interfaces.

In much the same way as plural protocol stacks increase the flexibility of a networked input device, plural protocol stacks 42 increase the flexibility of a networked output device such as the printer, by allowing the NIC to communicate via the network interface using plural different output protocols. Again, in FIG. 4, two different protocol stacks are shown including IP protocol stack 48 and IPX protocol stack 49. Other protocol stacks may also be provided; and it is further possible for the networked output device to operate over even a single protocol stack.

In the FIG. 1 example, printer control application 44, JETSEND interaction protocol 45 and JETSEND agent 46 are all shown as operating over the IP protocol stack 48. It is to be understood that this is representative only, and it is typical for printer control applications, JETSEND interface protocols, and JETSEND agents also to operate over each of the different protocol stacks included in the NIC.

January 2000

- [Data Memory Paging Management](#) by Hugh O'Byrne
- [Slow and Steady Never Lost the Race](#) by Michael Barr
- [Embedding TCP/IP](#) by Thomas Herbert
- [An Introduction to the JetSend Protocol](#) by John Meadows
- More on Overloading with const by Dan Saks
- What Barely Works by P.J. Plauger

February 2000

- [Protecting Binary Executables](#) by Matt Fisher
- [Embedding with GNU: The GNU Compiler and Linker](#) by Bill Gatliff
- [Data Memory Paging Management, Part 2](#) by Hugh O'Byrne
- Implementing Speech Coding Algorithms by Emmanuel Roy
- Embedding with XML by Scott Lawrence
- Top-Level cv-Qualifiers in Function Parameters by Dan Saks

March 2000

- [Modeling Complex Behavior Simply](#) by Stephen J. Mellor
- [How to Simplify Complexity](#) by Bran Selic
- [An Introduction to USB Development](#) by Jack Ganssle

April 2000

- [Safe Memory Utilization](#) by Niall Murphy
- [TCP/IP or Not TCP/IP?](#) by Michael Barr
- [Exposing MIB Data to a Web-based Interface](#) by Kedron Wolcott
- [Configuring TCP/IP Hosts, Methods, and Protocols](#) by Christopher K. Leidigh
- Passing Parameters by Value with const by Dan Saks

May 2000

- [Playback: Reality-Based System Testing](#) by Wes Howl
- A Generic **Protocol** Engine for Synchronous Protocols by Andrey Jivsov
- [A 'C' Test: The 0x10 Best Questions for Would-be Embedded Programmers](#) by Nigel Jones
- [Exposing MIB Data to a Web-based Interface, Part 2](#) by Kedron Wolcott

June 2000

- [Deadline Monotonic Analysis](#) by Ken Tindell
- [MAC Daddy](#) by Michael Barr
- [Infrared Communications with IrDA](#) by Charles Knutson
- Is an Array Really Just a Pointer? by Dan Saks

July 2000

- [Software-Based Memory Testing](#) by Michael Barr
- Design by Contract for C Programmers by Steve Kapp

- [Mid Year's Resolutions](#) by Michael Barr
- [Bluetooth Basics](#) by Rebecca Spaker
- [Arrays as Function Parameters](#) by Dan Saks

August 2000

- [State Oriented Programming](#) by Miro Samek and Paul Montgomery
- [Modeling Dynamic Systems](#) by Jim Ledin
- [Pardon Me, Do You Have the Time?](#) by David Hinerman
- [Sensible Software Testing](#) by Sean Beatty
- [Linux, Interrupted](#) by Thomas Besemer
- [Network Talk: Voice Over IP](#) by Yashvant Jani

September 2000

- [Sequence Enumeration](#) by Rob Oshana
- [Doing Hartley Smartly](#) by Robert Scott
- [Get by Without an RTOS](#) by Michael Melkonian
- [Configuration Management Tips](#) by Allan Folz
- [Internet Working](#) by Michael Barr
- [Numeric Literals](#) by Dan Saks

October 2000

- [Measure Twice, Cut Once](#) by John Fusco
- [PID \(Proportional, Integral, Derivative controller\); without a PhD](#) by Tim Wescott
- [Abstracting System Hardware for Maximum Reuse](#) by Joseph Lemieux
- [Strategies for Debugging Embedded Systems](#) by Gregory Eakman
- [HIDs Up](#) by Jan Axelsson
- [Character and String Literals](#) by Dan Saks

November 2000

- [Safety First: Avoid Software Mishaps](#) by Charles Knutson and Sam Carmichael
- [Eight-Bit OO \(Hard Eight\)](#) by Marshall Meier
- [Better Than Average](#) by Phil Ekstrom
- [Watchdog Timers](#) by Niall Murphy
- [Distributed Software Design: Challenges and Solutions](#) by Bran Selic
- [Network Protocols for the Home](#) by John Canosa

December 2000

- [Embedded State Machine Implementation](#) by Martin Gomez
- [Software Components for Real Time](#) by David B. Stewart
- [On the ROPES](#) by Bruce Powell Douglas
- [Flexible Dynamic Array Allocation](#) by Richard Hogaboom
- [How to Start a Consulting Business](#) by Bob Zeidman
- [Tiny File System](#) by Ed Sutter

January 2001

- [Testing Graphical Applications](#) by David Fell
- [Linux Porting Guide](#) by Rajesh Palani
- [Real-Time Ethernet](#) by Joe Kerkes
- [One Cheap Network Topology](#) by H. Michael Willey
- [An Introduction to References](#) by Dan Saks

February 2001

- [Object-Oriented C: Creating Foundation Classes Part 1](#) by Matthew Curreri
- [Context Switch](#) by David Kalinsky
- [VLIW Architecture Emerges as Embedded Alternative](#) by Alexander Wolfe
- [References and const](#) by Dan Saks

March 2001

- [Dynamic System Simulation](#) by Jim Ledin
- [Deadline Scheduling](#) by Peter Dibble
- [Object-Oriented C: Creating Application Classes](#) by Matthew Curreri

April 2001

- [Fixed-Point Math in C](#) by Joe Lemieux
- [Automatic Units Tracking](#) by Christopher Rettig
- [POSIX in Real-Time](#) by Kevin M. Obenland
- [Assertiveness Training for Timid Programmers](#) by Niall Murphy
- [Queueing Theory for Dummies](#) by David Kalinsky
- [Reentrancy](#) by Jack G. Ganssle
- [References vs. Pointers](#) by Dan Saks

May 2001

- [Analog-to-Digital Converters](#) by Stuart Ball
- [A Better Way to Process Messages](#) by Bill Trudell
- [Assert Yourself](#) by Niall Murphy
- [Real-Time Linux](#) by Alex Ivchenko
- [Memory Types](#) by Michael Barr
- [A New Appreciation for Data Types](#) by Dan Saks

June 2001

- [Verification and Validation](#) by Charles Knutson and Sam Carmichael
- [Application Code and RTLinux](#) by Alex Ivchenko
- [Understanding Interrupts](#) by Russell Massey
- [Lvalues and Rvalues](#) by Dan Saks

July 2001

- [Debug Tip: Circular History Buffer](#) by Andy Purcell
- [A Version Therapy](#) by Niall Murphy
- [Mobile IP](#) by Larry Mittag

- [Volatile](#) by Nigel Jones
- [Non-modifiable Lvalues](#) by Dan Saks
- [Asynchronicity](#) by Jack Ganssle

August 2001

- [Managing Changing Requirements](#) by Stephen J. Mellor
- [Interrupts in C++](#) by Alan Dorfmeier and Pat Baird
- [Introduction to I²C](#) by David Kalinsky and Roee Kalinsky
- [Shades of Gray](#) by Don Morgan
- [Metastability and Firmware](#) by Jack G. Ganssle

September 2001

- [Safety Critical Operating Systems](#) by David Kleidermacher and Mark Griglock
- [Software-Friendly Hardware](#) by Christopher Leddy
- [Drawbacks of Object-Oriented Programming for Embedded Systems](#) by Niall Murphy
- [Pulse Width Modulation](#) by Michael Barr
- [Reference Initializations](#) by Dan Saks

October 2001

- [C++ on Low-End Micros](#) by Jeffrey Kohler
- [Embed with the Mailman](#) by Tim Jones
- [Watchdog Timers](#) by Niall Murphy and Michael Barr
- [Passing by Reference-to-const](#) by Dan Saks
- [Interrupt Latency](#) by Jack G. Ganssle

November 2001

- [Capturing Real-Time Requirements](#) by Bruce Powel Douglass
- [Reentrancy in Protocol Stacks](#) by T. Sridhar
- [Real Time](#) by David B. Stewart
- [Symbolic Constants](#) by Dan Saks

December 2001

- [Modular Programming in C](#) by John R. Hayes
- [Hardware-in-the-Loop Simulation](#) by Martin Gomez
- [Twiddle Bits](#) by Michael Gauland
- [Enumeration Constants vs. Constant Objects](#) by Dan Saks

January 2002

- [Embedding with GNU: Newlib](#) by Bill Gatliff
- [Embedding with GNU: Newlib, Part 2](#) by Bill Gatliff
- [Low-Power Design](#) by Mike Willey and Kris Stafford
- [Endianness](#) by Christopher Brown and Michael Barr
- [Constant Objects and Constant Expressions](#) by Dan Saks

February 2002

- [Temperature Measurement Techniques](#) by Stuart Ball
- [Serial Peripheral Interface](#) by David Kalinsky and Roe Kalinsky
- [Symbolic Constant Expressions](#) by Dan Saks

March 2002

- [Bug Fishing](#) by Mark Lambuth
- [Flushing Out Memory Leaks](#) by Niall Murphy
- [Taking Out the Garbage](#) by Kelvin Nilsen
- [Rate Monotonic Scheduling](#) by David B. Stewart and Michael Barr

April 2002

- [Simulation Takes Off with Hardware](#) by Jim Ledin
- [More On Memory Leaks](#) by Niall Murphy
- [USB Debug Tips](#) by Jan Axelson
- [Priority Inversion](#) by David Kalinsky and Michael Barr
- [As Precise as Possible](#) by Dan Saks

April 2002

- [Object-Oriented State Machines](#) by Eric Kavins and Uluc Saranli

moria.greycastle.net

webmaster@moria.iwarp.com

Last modified: 2002-Jun-09T15:40:40 PDT

This page generated using [Genpage](#) - Version: 1.0.7



This is Google's cache of <http://morja.greycastle.net/magazines/esp.html>.

Google's cache is the snapshot that we took of the page as we crawled the web.

The page may have changed since that time. Click here for the [current page](#) without highlighting.

To link to or bookmark this page, use the following url: <http://www.google.com/search?q=cache:7vM1JJdkTtMC:morja.greycastle.net/magazines/esp.html+John+Meadows+an+introduction+to+the+jetsend+protocol++++&hl=8>

Google is not affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **john meadows introduction jetsend protocol**

Embedded Systems Programming

This is the best magazine that I subscribe to. If you work with embedded systems then this is *the* magazine to get. The web page has lots of other useful information as well.

ESP has a [article archive](#) but not all the articles from every issue is linked for issues prior to September 2000. I believe this is because you can [buy a CD-ROM](#) with all the articles back to 1988. You can also [search through the archive](#) by topic or author.

Articles

January 1999

- [UML Statecharts](#) by Bruce Powel Douglass
- [Effective C++ Memory Allocation](#) by Aaron Dailey

February 1999

- [Scheduling a Real-Time Program](#) by R.C. Lacovara
- [Hardware-in-the-Loop Simulation](#) by Jim A. Ledin
- [Const T vs T Const](#) by Dan Saks

March 1999

- [Time Triggered Protocol: TTP/C](#) by Ross Bannatyne
- [Selecting a Real-Time Operating System](#) by Greg Hawley
- [Ensuring Static Initialization in C++](#) by Dan Saks

April 1999

- [PIC Programming in C Using Hand Compilation](#) by John Hilton

May 1999

- [Navigating through New Development Environments](#) by Jack G. Ganssle
- [Arrays of Pointers to Functions](#) by Nigel Jones
- [A Protocol for GPS Communications](#) by Jeff Stefan

- [Function Name Overloading](#) by Dan Saks

June 1999

- [Firmware Development on a Virtual Target](#) by C. Larry Rogers
- [Fundamentals of Firewire](#) by John Canosa
- [Linking Function Calls](#) by Dan Saks
- [What's in a Namespace?](#) by P.J. Plauger

July 1999

- [GUI Development: Embedding Graphics, Part I](#) by Niall Murphy
- [A Generic API for Bit Manipulation in C](#) by Richard Hogaboom
- [Sequence Points and Parallelism](#) by P.J. Plauger

August 1999

- [GUI Development: Embedding Graphics, Part II](#) by Niall Murphy
- [Controlling the Transmit Enable Line on RS-485 Transceivers](#) by Nigel Jones
- [function Signatures and Name Mangling](#) by Dan Saks

September 1999

- [Embedding with GNU: The GNU Debugger](#) by Bill Gatliff
- [Sizing Throughput Requirements on Real-Time Systems](#) by Don Bruyère
- [In Praise of the #error Directive](#) by Nigel Jones
- [Architecting Embedded Systems for Add-on Software Modules](#) by Michael Barr
- [A Driver-Based Approach to Protocol Stack Design](#) by Curt Schwaderer
- [Using const and volatile in Parameter Types](#) by Dan Saks

October 1999

- [30 Pitfalls for Real-Time Software Developers, Part 1](#) by David B. Stewart
- [Safety-Critical Embedded Systems](#) by Bruce Powel Douglass
- [Auto-Configuration of Embedded Systems](#) by Changrong Li

November 1999

- [Common Architectures for Communications](#) by David Nix
- [More Pitfalls for Real-Time Software Developers](#) by David B. Stewart
- [Embedding with GNU: The gdb Remote Serial Protocol](#) by Bill Gatliff

December 1999

- [Rules for Defensive C Programming](#) by Dinu Madau
- [Using Flash in an 8051-Based System](#) by Tony Gray
- [How to Reduce State Machine Complexity](#) by Charles E. Hine
- [Introduction to TCP/IP](#) by Thomas Herbert
- [Overloading with const](#) by Dan Saks
- [Lying Code](#) by P.J. Plauger